

# Outline

- ◇ Best-first search
- ◇ A\* search
- ◇ Heuristics

## Review: Tree search

```
function TREE-SEARCH(problem, fringe) returns a solution, or failure
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST[problem] applied to STATE(node) succeeds return node
    fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

A strategy is defined by picking the **order of node expansion**

## Best-first search

**Idea:** use an **evaluation function** for each node  
– estimate of “desirability”

⇒ Expand most desirable unexpanded node

**Implementation:**

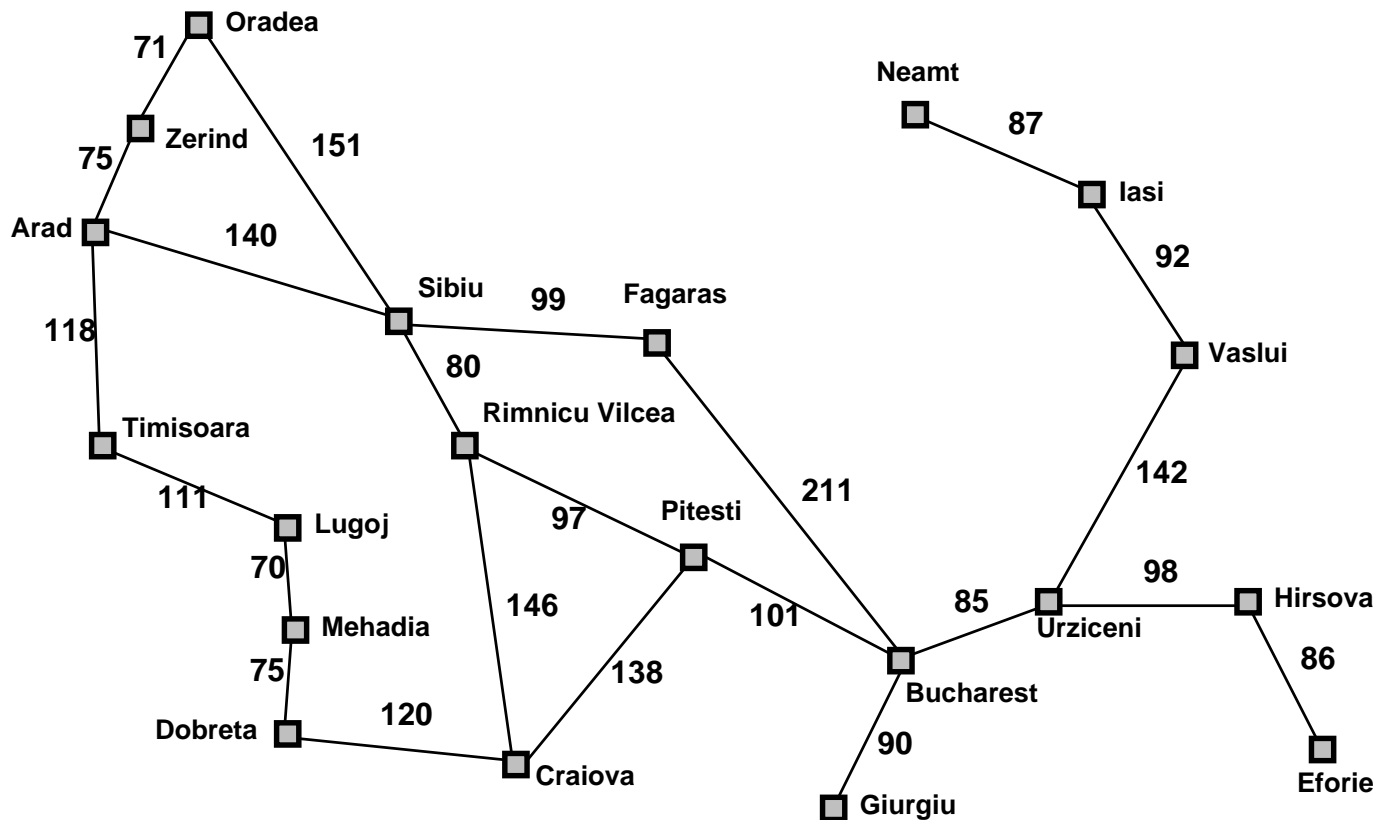
*fringe* is a queue sorted in decreasing order of desirability

Special cases:

greedy search

A\* search

# Romania with step costs in km



Straight-line distance  
to Bucharest

<b>Arad</b>	366
<b>Bucharest</b>	0
<b>Craiova</b>	160
<b>Dobreta</b>	242
<b>Eforie</b>	161
<b>Fagaras</b>	178
<b>Giurgiu</b>	77
<b>Hirsova</b>	151
<b>Iasi</b>	226
<b>Lugoj</b>	244
<b>Mehadia</b>	241
<b>Neamt</b>	234
<b>Oradea</b>	380
<b>Pitesti</b>	98
<b>Rimnicu Vilcea</b>	193
<b>Sibiu</b>	253
<b>Timisoara</b>	329
<b>Urziceni</b>	80
<b>Vaslui</b>	199
<b>Zerind</b>	374

## Greedy search

Evaluation function  $h(n)$  (**h**euristic)

= estimate of cost from  $n$  to the closest goal

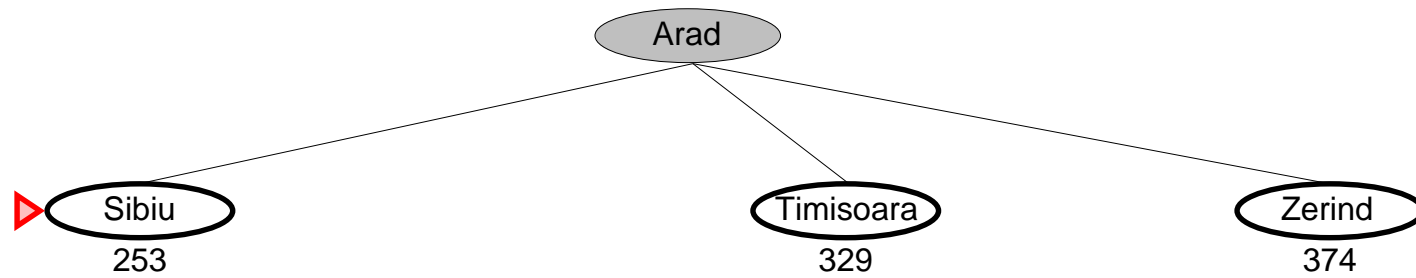
E.g.,  $h_{\text{SLD}}(n)$  = straight-line distance from  $n$  to Bucharest

Greedy search expands the node that **appears** to be closest to goal

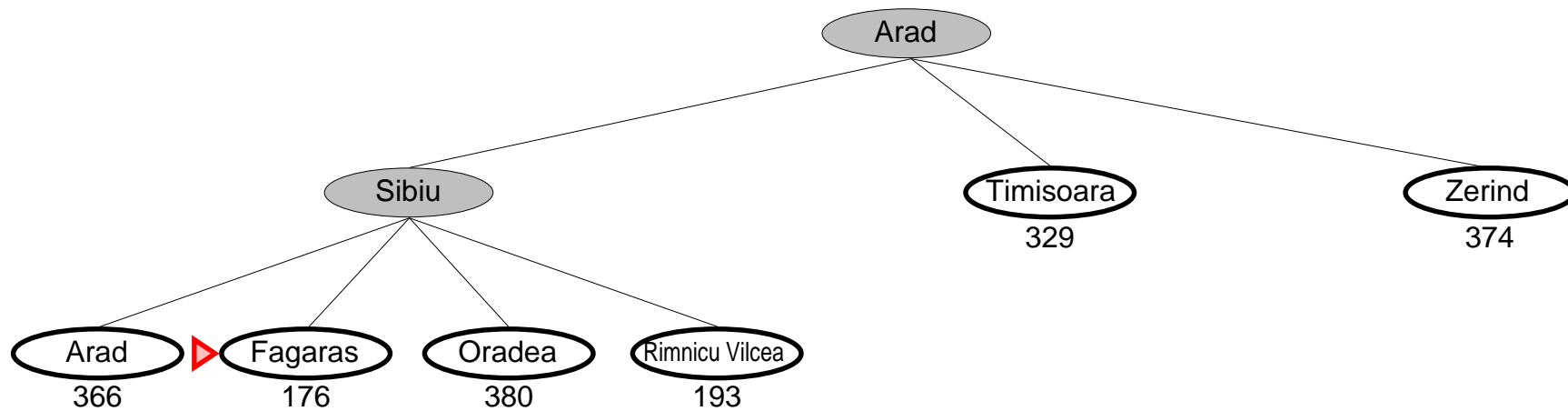
## Greedy search example

▶ Arad  
366

## Greedy search example

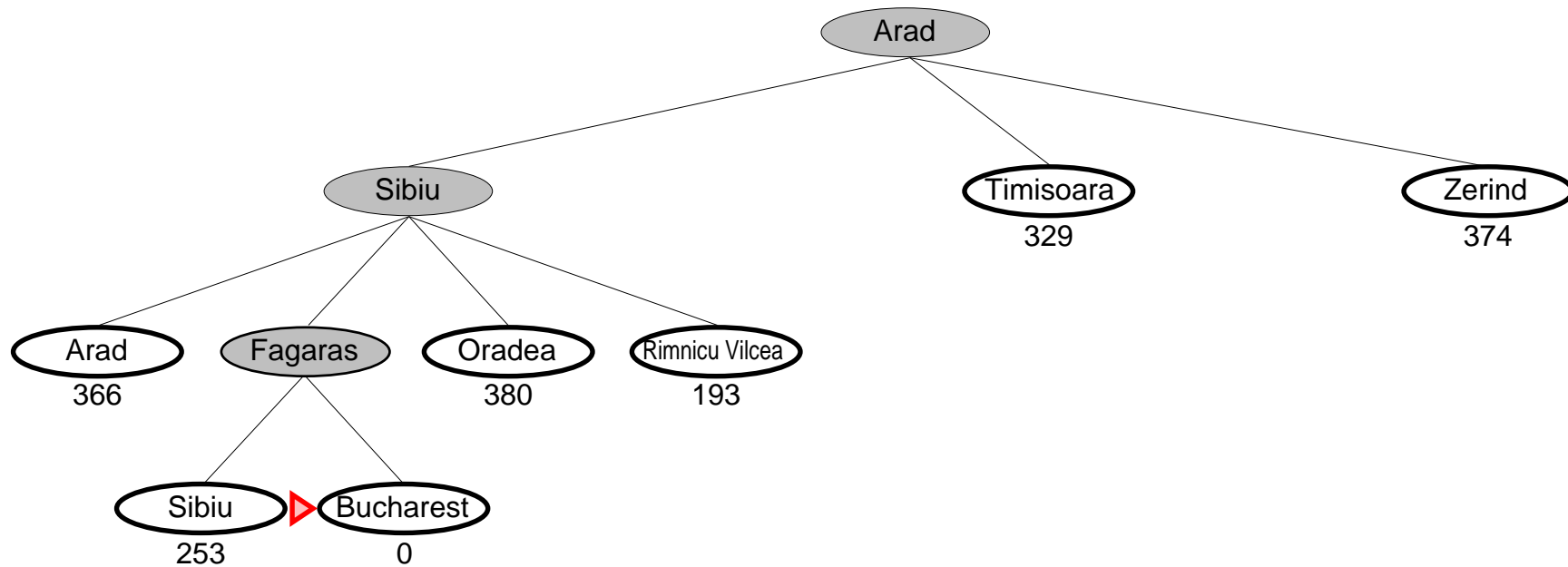


# Greedy search example





# Greedy search example



# Properties of greedy search

Complete??

## Properties of greedy search

Complete?? No—can get stuck in loops, e.g., with Oradea as goal,

lası → Neamt → lası → Neamt →

Complete in finite space with repeated-state checking

Time??

## Properties of greedy search

Complete?? No—can get stuck in loops, e.g.,

lasi  $\rightarrow$  Neamt  $\rightarrow$  lasi  $\rightarrow$  Neamt  $\rightarrow$

Complete in finite space with repeated-state checking

Time??  $O(b^m)$ , but a good heuristic can give dramatic improvement

Space??

## Properties of greedy search

Complete?? No—can get stuck in loops, e.g.,

lasi  $\rightarrow$  Neamt  $\rightarrow$  lasi  $\rightarrow$  Neamt  $\rightarrow$

Complete in finite space with repeated-state checking

Time??  $O(b^m)$ , but a good heuristic can give dramatic improvement

Space??  $O(b^m)$ —keeps all nodes in memory

Optimal??

## Properties of greedy search

Complete?? No—can get stuck in loops, e.g.,

lasi  $\rightarrow$  Neamt  $\rightarrow$  lasi  $\rightarrow$  Neamt  $\rightarrow$

Complete in finite space with repeated-state checking

Time??  $O(b^m)$ , but a good heuristic can give dramatic improvement

Space??  $O(b^m)$ —keeps all nodes in memory

Optimal?? No

## A\* search

Idea: avoid expanding paths that are already expensive

Evaluation function  $f(n) = g(n) + h(n)$

$g(n)$  = cost so far to reach  $n$

$h(n)$  = estimated cost to goal from  $n$

$f(n)$  = estimated total cost of path through  $n$  to goal

A\* search uses an **admissible** heuristic

i.e.,  $h(n) \leq h^*(n)$  where  $h^*(n)$  is the **true** cost from  $n$ .

(Also require  $h(n) \geq 0$ , so  $h(G) = 0$  for any goal  $G$ .)

E.g.,  $h_{\text{SLD}}(n)$  never overestimates the actual road distance

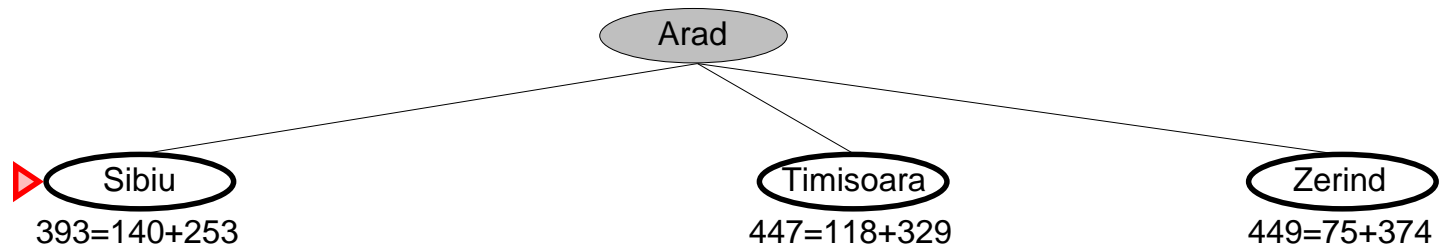
**Theorem:** A\* search is optimal

## $A^*$ search example

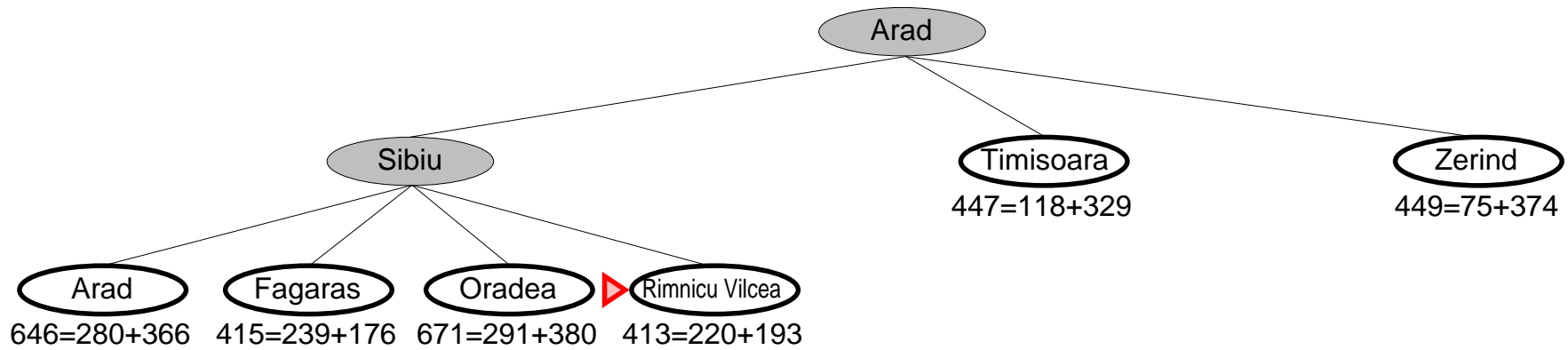
▶ Arad  
 $366=0+366$



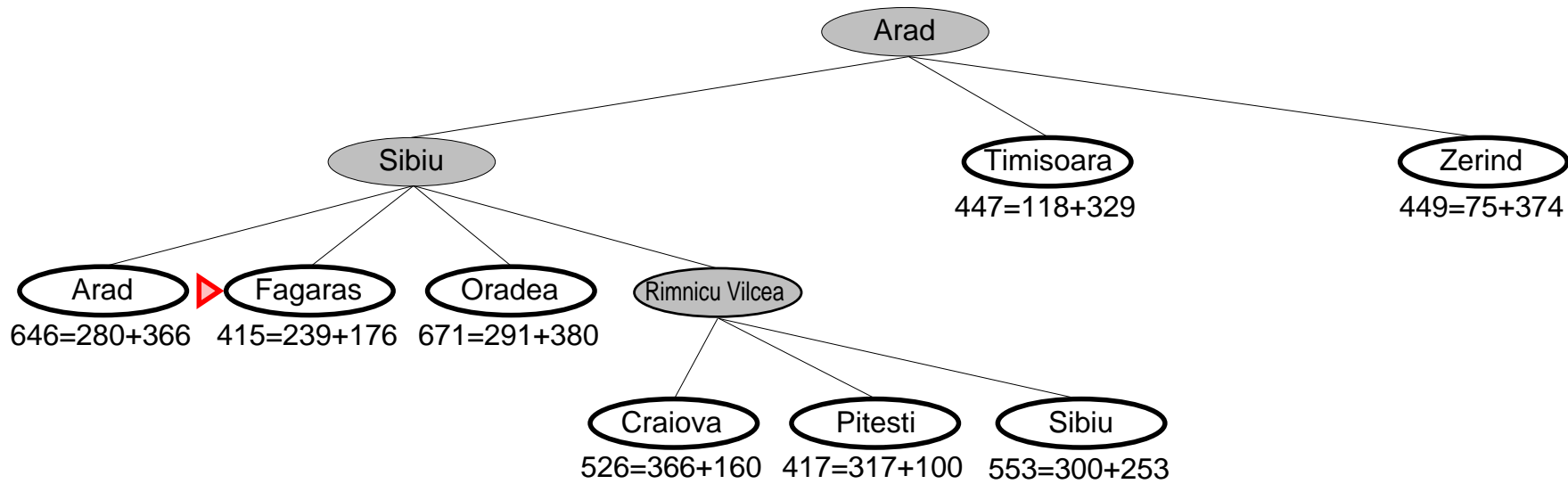
## A\* search example



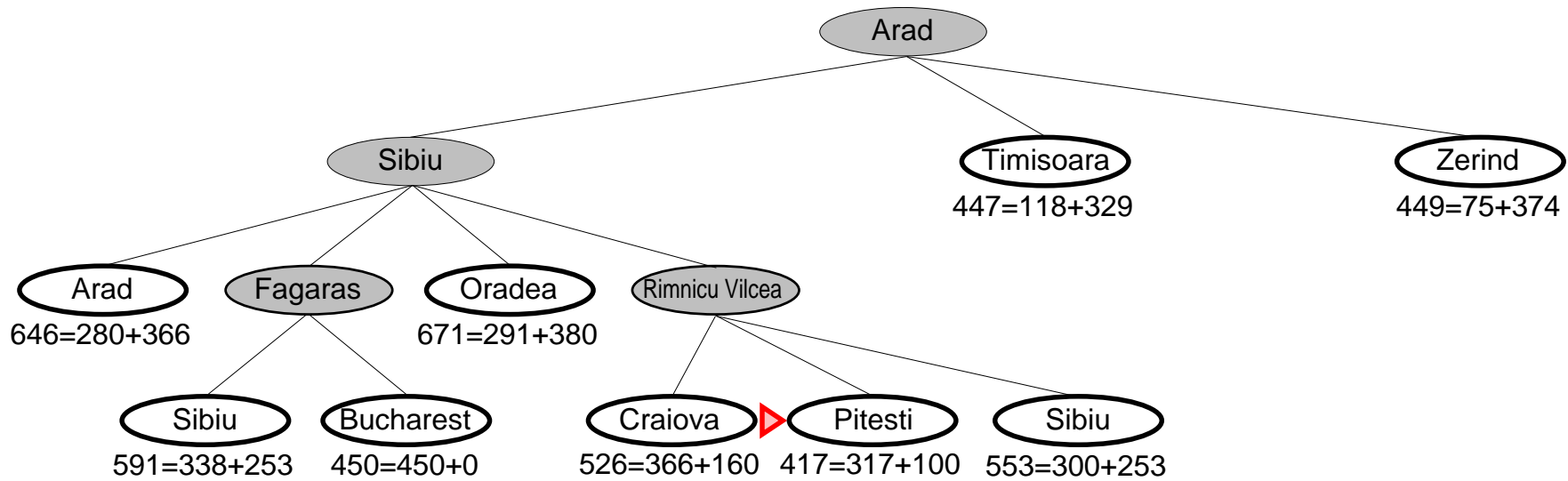
## A\* search example



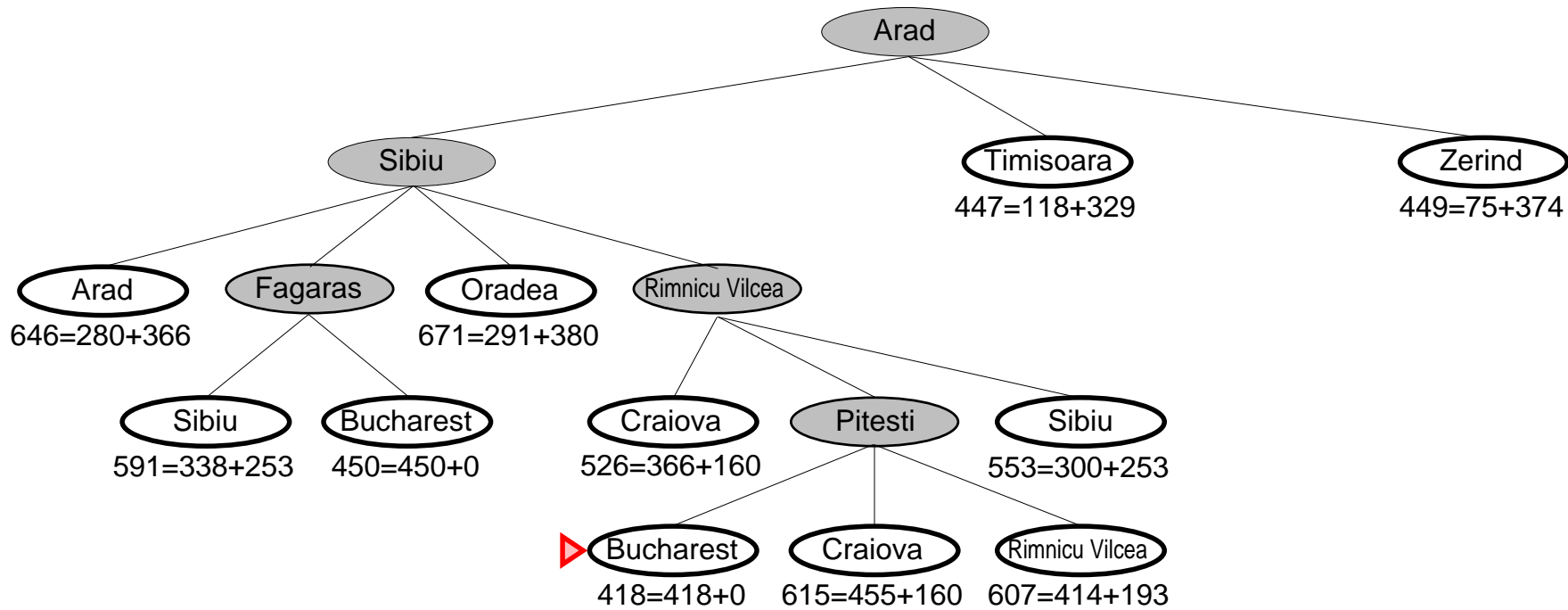
## A\* search example



## A\* search example

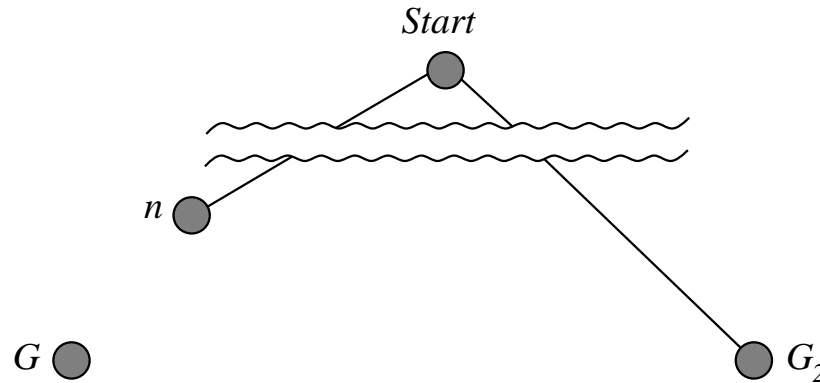


## A\* search example



## Optimality of $A^*$ (standard proof)

Suppose some suboptimal goal  $G_2$  has been generated and is in the queue. Let  $n$  be an unexpanded node on a shortest path to an optimal goal  $G_1$ .



$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible} \end{aligned}$$

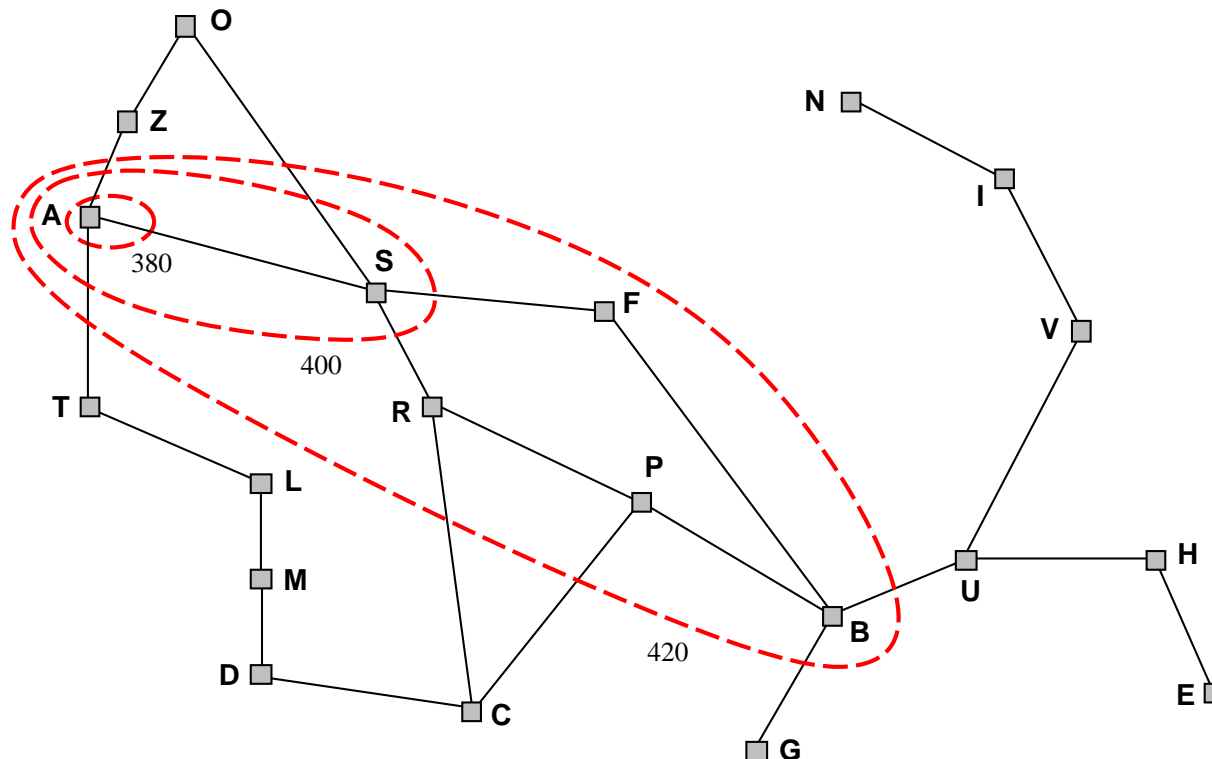
Since  $f(G_2) > f(n)$ ,  $A^*$  will never select  $G_2$  for expansion

## Optimality of $A^*$ (more useful)

**Lemma:**  $A^*$  expands nodes in order of increasing  $f$  value\*

Gradually adds “ $f$ -contours” of nodes (cf. breadth-first adds layers)

Contour  $i$  has all nodes with  $f = f_i$ , where  $f_i < f_{i+1}$



# Properties of $A^*$

Complete??



## Properties of $A^*$

Complete?? Yes, unless there are infinitely many nodes with  $f \leq f(G)$

Time??

## Properties of $A^*$

Complete?? Yes, unless there are infinitely many nodes with  $f \leq f(G)$

Time?? Exponential in [relative error in  $h \times$  length of soln.]

Space??

## Properties of $A^*$

Complete?? Yes, unless there are infinitely many nodes with  $f \leq f(G)$

Time?? Exponential in [relative error in  $h \times$  length of soln.]

Space?? Keeps all nodes in memory

Optimal??

## Properties of $A^*$

Complete?? Yes, unless there are infinitely many nodes with  $f \leq f(G)$

Time?? Exponential in [relative error in  $h \times$  length of soln.]

Space?? Keeps all nodes in memory

Optimal?? Yes—cannot expand  $f_{i+1}$  until  $f_i$  is finished

$A^*$  expands all nodes with  $f(n) < C^*$

$A^*$  expands some nodes with  $f(n) = C^*$

$A^*$  expands no nodes with  $f(n) > C^*$